Microsoft DevOps Solutions: Developing a Modern Source Control Strategy

DEVELOPING A MODERN SOURCE CONTROL STRATEGY

Marcel de Vries
CTO

@marcelv https://Fluentbytes.com

Exam Objectives Covered in This Course

Develop a modern source control strategy

Plan and implement branching strategies for the source code

Configure repositories

Outline

What is considered a modern source control strategy?

Migrating to GitHub or Azure DevOps

Manage and store large files in Git

Cross repository sharing

Implement workflow hooks

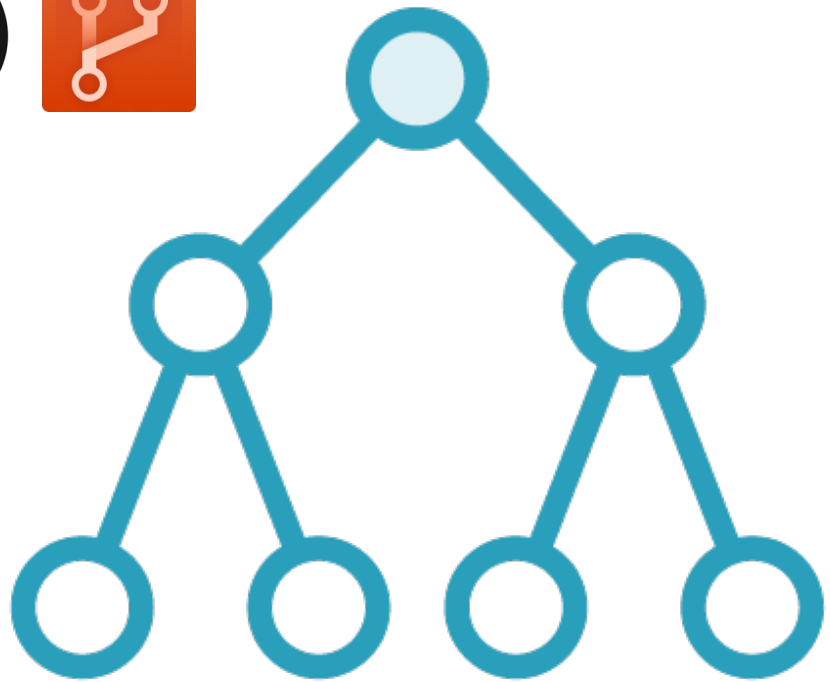
Summary

What Is Considered a Modern Source Control Strategy?

Types of Source Control Systems

Centralized

Distributed



https://en.wikipedia.org/wiki/Comparison_of_version_control_software

Centralized Source Control

e.g. SVN, PVCS, Source Safe and Team Foundation Server

Strengths

Scales to very large codebases

Fine level permission control

Allows usage monitoring

Ability to lock files exclusively

Best used for

Large integrated codebases

Control and auditability over source code down to the file level

When codebase has hard to merge file types

Distributed Source Control

Mercurial, Git

Strengths

Full offline experience & Speed

Complete repository with portable history

Cross platform Support

Growing usage in the market

Pull requests for code review

Best used for

Modular codebases

Integrating with open source

Highly distributed teams

Portable codebases between platforms

New codebases

Migrating to GitHub or Azure DevOps

Planning Considerations

Do you need full history migration?

- Tip migration often enough!
- Add breadcrumb files to point to old system

Enable Git Credential Manager

Delete files or data that binds your code to the legacy version control system

- $tfs, .svn

Adding Important Git Files

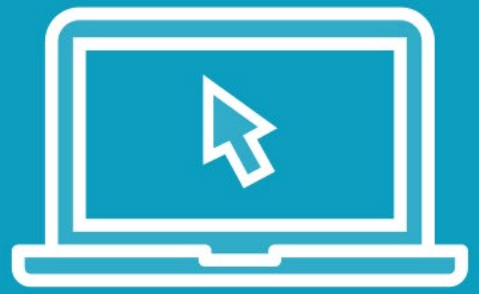
Convert version control system-specific directives

- .gitignore
- .gitattributes

Don't invent your own files, use what is already available

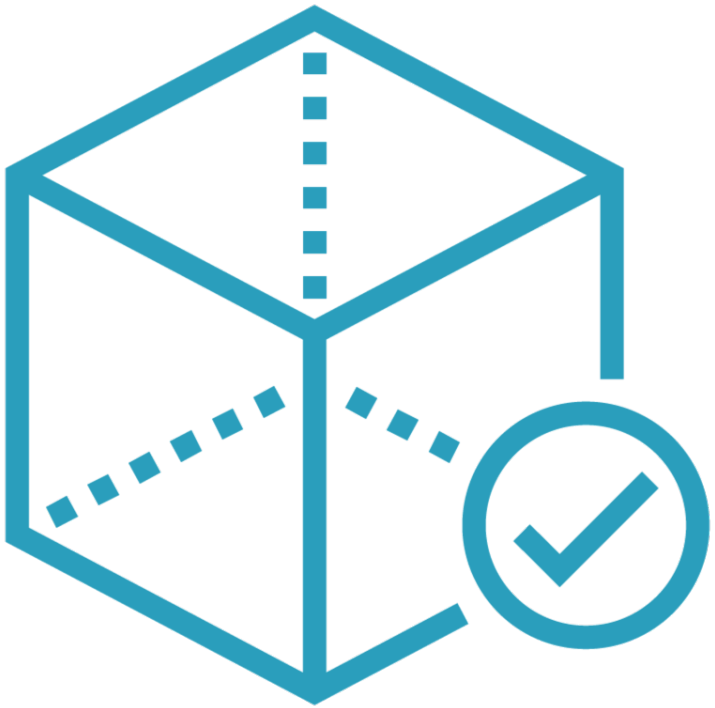
- https://github.com/github/gitignore

Demo



Setting up your .gitignore file

Out of the Box Supported Migrations



GitHub supports out of the box migrations

- Git
- Subversion
- Mercurial
- TFS

Azure DevOps supported migrations

- TFS server
- Azure DevOps Server

Git Sub Modules and Sub Trees

Both are a way to share code between repos without duplicating code

A subtree is a copy of a repository that is pulled into a parent repository

- Easier to pull and harder to push

A submodule is a pointer to a specific commit in another repository

- Easier to push but harder to pull

Manage and Store Large Files in Git

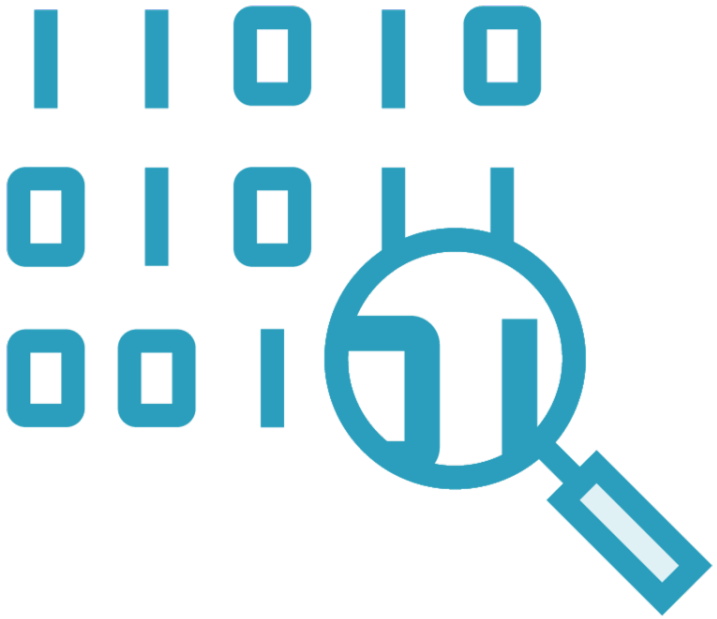
Files You Should Not Commit

Don’t commit outputs

Don’t commit large frequently changed binary assets

Don’t commit compressed archives

Handling Large Files (Git LFS)



Used when you can't discard large files and need to add them to your repo

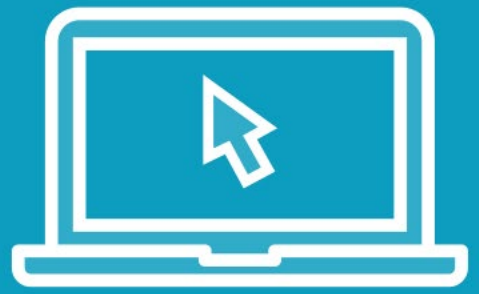
Keeps the standard Git workflow

- No limitations in size

Limitations

- Every Git client used by your team must install the Git LFS client
- No merge possible on binary files

Demo



How to Use Setup and Use Git LFS

Cross Repository Sharing

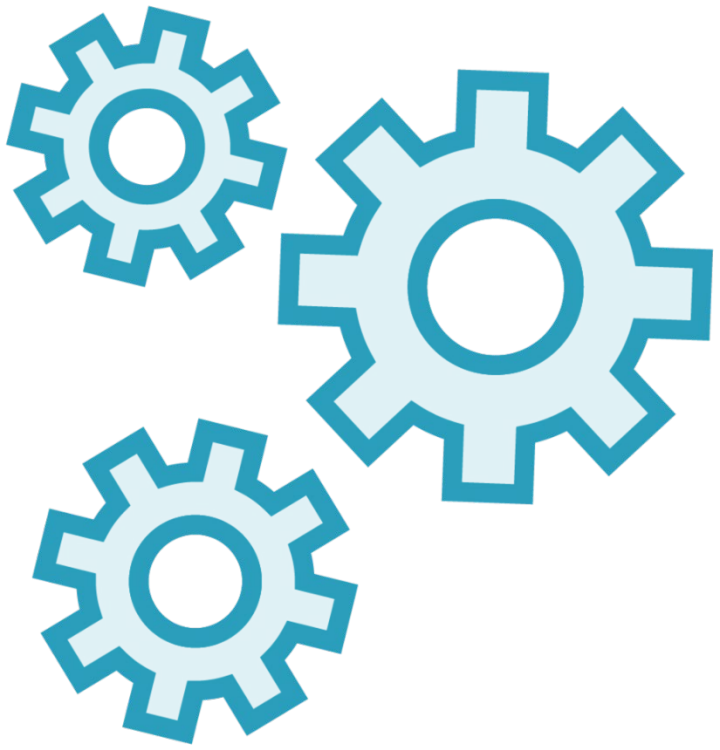
Package Management

Cross repository sharing of software artifacts

Various package management solutions

- Node Package Manager (NPM)
- Maven
- NuGet
- Ruby Gems
- Etc.

Handling the Dependency Supply Chain



Package management creates a supply chain of software outside of your organization

Use a package management solution

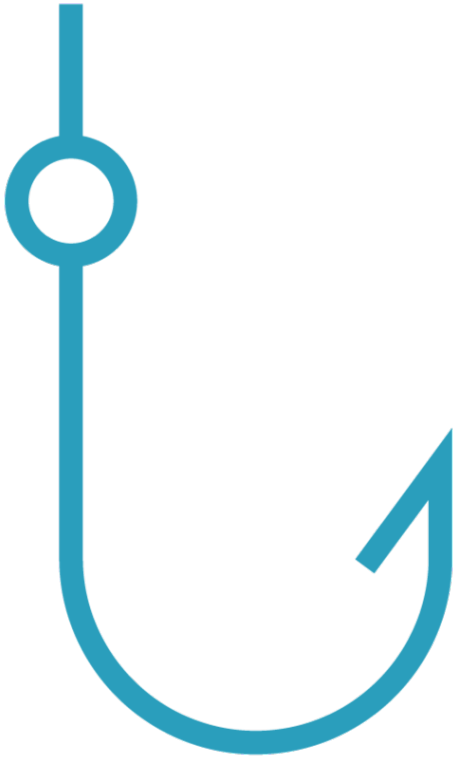
- GitHub, Azure DevOps Artifacts
- Upstream proxy

Keep versions up to date

- NuKeeper
- Dependabot

Implement Workflow Hooks

Using Web Hooks



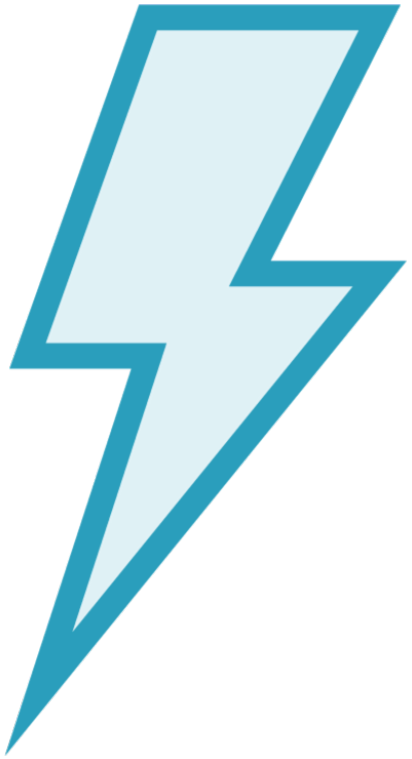
Azure DevOps

- Sends SOAP request on specified event
  - Build, Repo, Extensions, Pipelines, Releases, Work

GitHub

- Sends HTTP Post to registered endpoint
  - Issues, Repo, Actions, Security

Using GitHub Actions



GitHub actions makes it easy to automate all your software workflows

- Run a workflow on any GitHub event

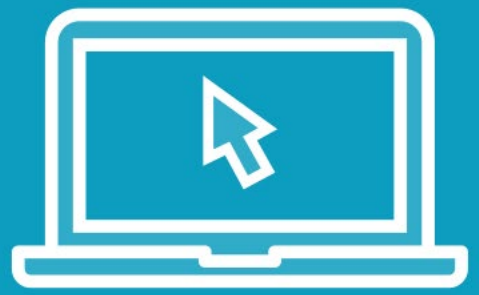
YAML Definition

CI/CD workflows

Pull request validations

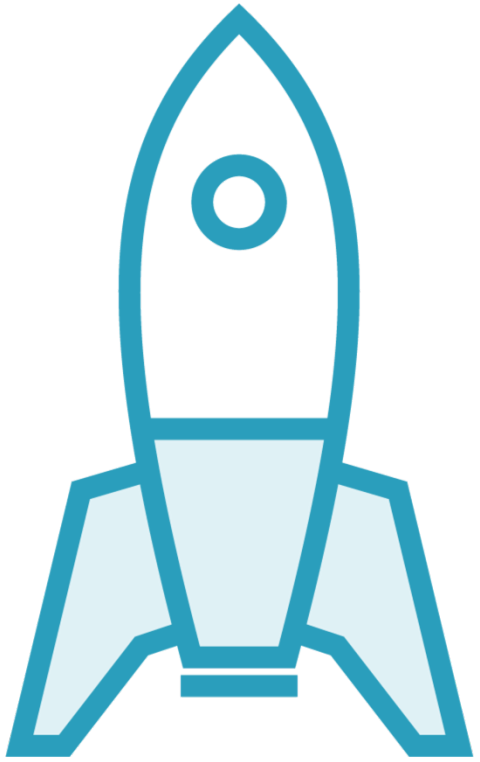
Security scanning

Demo



Implementing CI/CD with GitHub Actions

Using Azure DevOps Yaml Pipelines



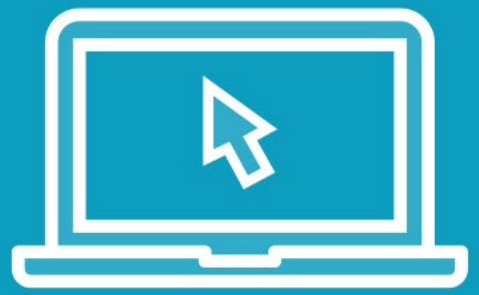
Azure pipelines is a cloud service

- Automatically build and test your code
- Works with just about any language or project type.

Supports multiple source control systems

- Git, GitHub, Sub Version, BitBucket, TFVC

Demo



Implementing CI/CD with Azure DevOps

Summary

What is considered a modern source control strategy?

Migrating to GitHub or Azure DevOps

Manage and store large files in Git

Cross repository sharing

Implement Workflow Hooks